

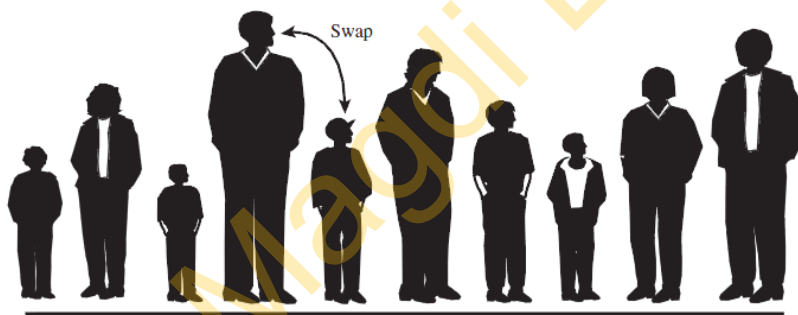
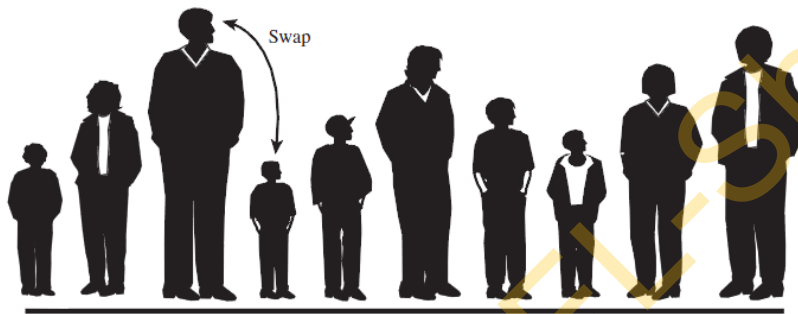
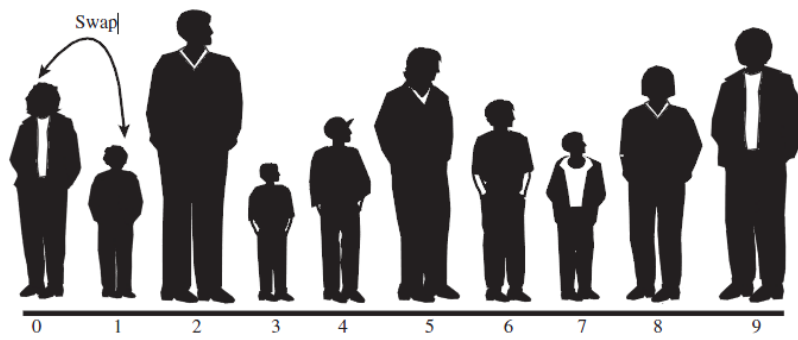
# Elementary sorting algorithms



## 1- Bubble Sort

The bubble sort is slow, but it's conceptually the simplest of the sorting algorithms

1. Compare two players.
2. If the one on the left is taller, swap them.
3. Move one position right.

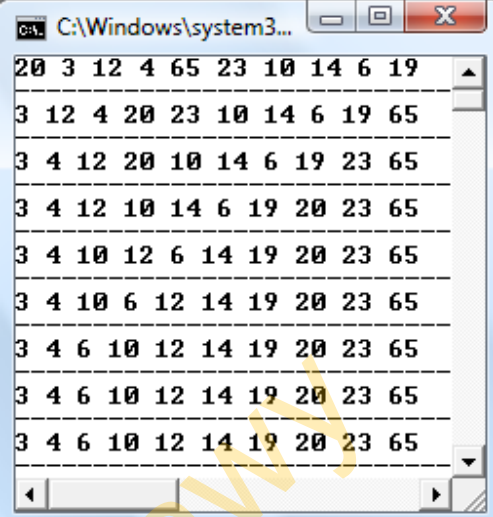


Sorted

```

void swap(int & , int &);
void main()
{
    int array1[10]={20,3,12,4,65,23,10,14,6,19};
    int size=10;
    for(int out = size-1; out>1; out--)
    {
        for(int in=0;in<out;in++)
        {
            if(array1[in] > array1[in + 1])
                swap(array1[in], array1[in + 1]);
        }
    }
}
void swap (int &x, int &y)
{
    int temp=x;
    x=y;
    y=temp;
}

```



20	3	12	4	65	23	10	14	6	19
3	12	4	20	23	10	14	6	19	65
3	4	12	20	10	14	6	19	23	65
3	4	12	10	14	6	19	20	23	65
3	4	10	12	6	14	19	20	23	65
3	4	10	6	12	14	19	20	23	65
3	4	6	10	12	14	19	20	23	65
3	4	6	10	12	14	19	20	23	65
3	4	6	10	12	14	19	20	23	65
3	4	6	10	12	14	19	20	23	65

البرنامج بعد أضافة خاصية الطباعة

```

#include "stdafx.h"
#include <iostream>
using namespace std;
int array1[10]={20,3,12,4,65,23,10,14,6,19};
void display();
void swap(int & , int &);
void main()
{
    int size=10;
    display();
    for(int out = size-1; out>1; out--)
    {
        for(int in=0;in<out;in++)
        {
            if(array1[in] > array1[in + 1])
                swap(array1[in], array1[in + 1]);
        }
        display();
    }
}
void swap (int &x, int &y)
{
    int temp=x;
    x=y;
    y=temp;
}
void display()
{
    for(int i=0;i<10;i++)
        cout << array1[i] << " ";
    cout << "\n" << "-----\n";
}

```

## Efficiency of the Bubble Sort

As you can see by watching the BubbleSort Workshop applet with 10 bars, the inner and inner+1 arrows make nine comparisons on the first pass, eight on the second, and so on, down to one comparison on the last pass. For 10 items, this is

$$9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 = 45$$

In general, where  $N$  is the number of items in the array, there are  $N-1$  comparisons on the first pass,  $N-2$  on the second, and so on. The formula for the sum of such a series is

$$(N-1) + (N-2) + (N-3) + \dots + 1 = N*(N-1)/2$$

$N*(N-1)/2$  is 45 ( $10*9/2$ ) when  $N$  is 10.

Thus, the algorithm makes about  $N^2/2$  comparisons (ignoring the  $-1$ , which doesn't make much difference, especially if  $N$  is large).

There are fewer swaps than there are comparisons because two bars are swapped only if they need to be. If the data is random, a swap is necessary about half the time, so there will be about  $N^2/4$  swaps. (Although in the worst case, with the initial data inversely sorted, a swap is necessary with every comparison.)

Both swaps and comparisons are proportional to  $N^2$ . Because constants don't count in Big O notation, we can ignore the 2 and the 4 and say that the bubble sort runs in  $O(N^2)$  time. This is slow, as you can verify by running the BubbleSort Workshop applet with 100 bars.